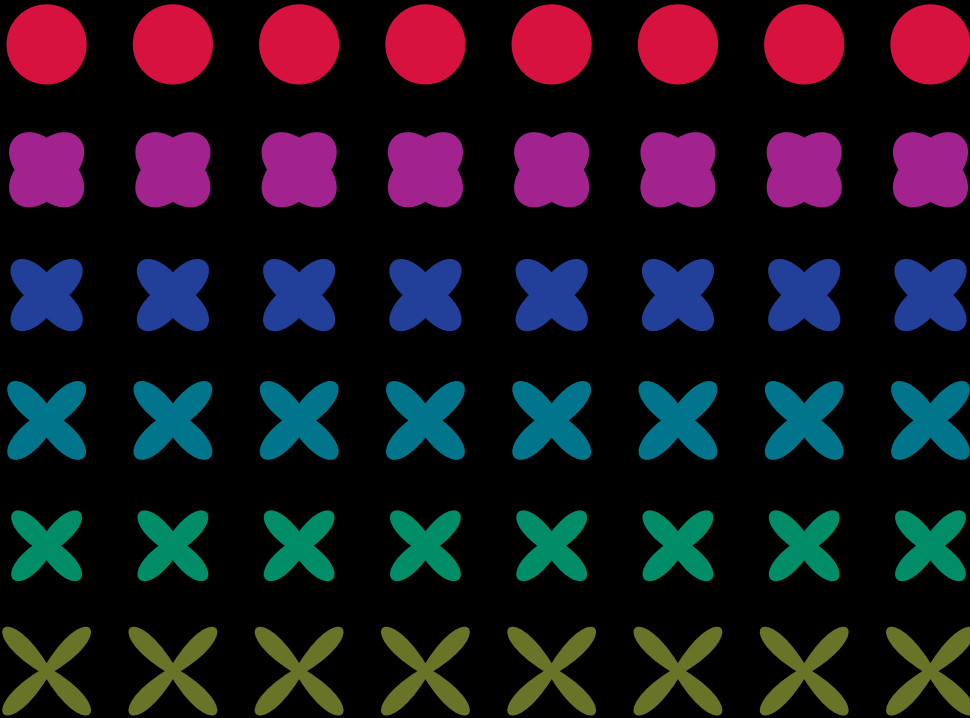


IUNA

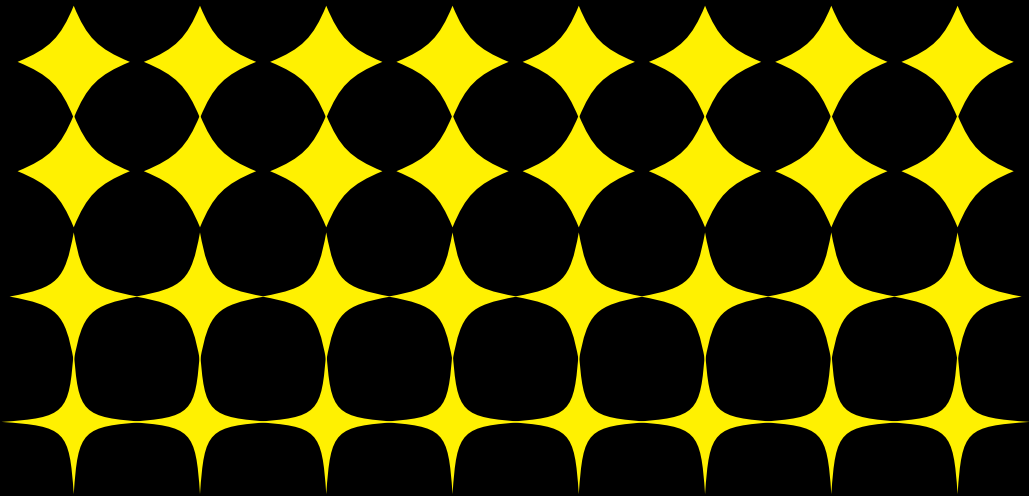
Instituto Universitario
Nacional del Arte

RIM

revista de investigación multimedia
año 3. número 3. marzo de 2011



3



IUNA

Instituto Universitario Nacional del Arte

editorial

... cabe tener en cuenta las posibilidades que pueden llegar a seguirse de los encuentros entre ciencia y las prácticas artísticas. En muchos casos, los artistas toman inspiración en los hallazgos de la ciencia, o investigan de manera creativa en algunos campos tecno-científicos (por ejemplo, y en estos momentos, la ingeniería genética, la nanotecnología, la matemática del límite, el software avanzado, la astronomía, la robótica, la inteligencia artificial, son campos en los que unos u otros artistas han encontrado inspiración directa y a partir de los que de hecho realizan alguna investigación creativa suficientemente relevante, cuando menos desde el punto de vista artístico).

(José Luis Brea, cultura_RAM, 2007)

En el año del Bicentenario (1810-2010) hacemos entrega del tercer número de la Revista de Investigación Multimedia del Área Transdepartamental de Artes Multimediales del Instituto Universitario Nacional del Arte.

La misma se presenta como una edición especial de los documentos de avance de la primera etapa del proyecto PICTO “Diseño y desarrollo de aplicaciones e interfases de Realidad Aumentada destinadas a síntesis y procesamiento de audio digital”, aprobado por la Agencia Nacional de Promoción Científica y Tecnológica - Fondo para la Investigación Científica y Tecnológica (Ministerio de Ciencia, Tecnología e Innovación Tecnológica - Presidencia de la Nación).

El proyecto parte del paradigma “continuo realidad-virtualidad” (Milgram, Kishino y Takemura, 1994) el cual ubica a los entornos virtuales en diversas categorías, cuyo rango se extiende desde la realidad física hasta la realidad virtual. Situados en este paradigma, entendemos por entornos de realidad aumentada a aquellos que logran conjugar elementos virtuales con la realidad física que nos rodea. Dentro de esta clasificación se encuadra la mayor parte de las experiencias de multimedia escénica, como las instalaciones, las *performances* y las intervenciones interactivas en las cuales realidad y virtualidad se funden.

En esta entrega volcamos los resultados de siete artículos que avanzan sobre los tópicos planteados. “Nuevas caracterizaciones de la actividad musical en el aula” de Prof. Carmelo Saitta abre un espacio de análisis sobre la vinculación música - espacio áulico. “Composición asistida en entorno PD” de los Dres. Pablo Cetta y Oscar Pablo Di Liscia, y “Medidas de similitud entre sucesiones ordenadas de grados cromáticos” del Dr. Oscar Pablo Di Liscia presentan técnicas de composición y organización del material musical en el entorno de composición en tiempo real *Pure data*. “Desarrollo de un sistema óptico para interfaces tangibles (mesa con pantalla reactiva)” y “Diseño de interface para el desarrollo de una pantalla sensible al tacto con aplicación musical” del Ing. Emiliano Causa exponen, respectivamente, el desarrollo de una mesa con pantalla sensible al tacto y el diseño de una interface de pantalla sensible al tacto (del tipo multitacto) aplicada a un editor gestual de música. “Técnicas de síntesis y procesamiento de sonido y su aplicación en tiempo real” del Lic. Matías Romero Costas repasa algunas de las técnicas de síntesis y procesamiento de sonido más utilizadas y difundidas, desde un punto de vista teórico, a través de ejemplos de aplicación, implementados en el entorno de programación Max-MSP. Por último, el artículo “Técnicas de programación vinculadas a la realidad aumentada y a las interfaces tangibles” del DCV Tarcisio Lucas Pirotta, presenta las principales funciones y bloques de programación disponibles en los paquetes de librería ARToolKit y reacTIVision, para su aplicación a proyectos de realidad aumentada e interfaces tangibles.

Edición especial dedicada al proyecto PICTO “Diseño y desarrollo de aplicaciones e interfases de Realidad Aumentada destinadas a síntesis y procesamiento de audio digital”, en el año del Bicentenario (1810-2010).

Comité Editorial de RIM

staff



Instituto Universitario Nacional del Arte

Rectora

Prof. Liliana Beatriz Demaio

Vicerrectora

Prof. Susana Pires Mateu

Secretaría General

Prof. María Martha Gigena

Secretaría de Asuntos Jurídico-Legales

Dra. Clara María Picasso Achaval

Secretaría de Asuntos Académicos

Prof. Oscar Steimberg

Secretaría de Investigación y Posgrado

Asesora Graciana Vázquez Villanueva

Secretaría de Extensión y Bienestar Estudiantil

Mg. Fernando Lerman

Secretaría de Desarrollo y Vinculación Institucional

Arq. Daniel Wolkowicz

Secretaría de Asuntos Económico-Financieros

Cont. Eduardo Jorge Auzmendi

Secretaría de Asuntos Administrativos

Asesor Federico Hernán Tessore

Secretaría de Infraestructura y Planeamiento

Arq. Nicolás Escobari



Área Transdepartamental de Artes Multimediales

Director

Prof. Carmelo Saitta

Secretario Académico

Dr. Pablo Cetta

Secretario Administrativo

Dr. Roberto Abait

Coordinación de Actividades de Investigación y Posgrado

Mg. Raúl Lacabanne

Coordinación de Actividades de Extensión y Bienestar Estudiantil

Prof. Gumersindo Jerónimo de Jesús Serrano Gómez

RIM

Director

Prof. Carmelo Saitta

Secretario de redacción

Dr. Pablo Cetta

Comité editorial

Ing. Emiliano Causa

Dr. Pablo Cetta

Dr. Pablo Di Liscia

Mg. Raúl Lacabanne

Prof. Gumersindo Jerónimo de Jesús Serrano Gómez

Arq. Daniel Wolkowicz

Colaboran en este número

Carmelo Saitta

Pablo Cetta

Pablo Di Liscia

Emiliano Causa

Matías Romero Costas

Tarcisio Lucas Pirotta

Diseño

Arq. Daniel Wolkowicz

Corrección de textos

Rossana Cabrera

RIM es una publicación del Área

Transdepartamental de Artes Multimediales
del IUNA

Yatay 843, Ciudad Autónoma de Buenos Aires,
República Argentina

Todos los derechos reservados

ISSN 1850-2954

Impreso en New Press Grupo Impresor S.A.

Paraguay 278, Avellaneda

Provincia de Buenos Aires

Marzo de 2011

Índice

Nuevas caracterizaciones de la actividad musical en el aula Carmelo Saitta	5
Composición asistida en entorno PD Dr. Pablo Cetta y Dr. Oscar Pablo Di Liscia	15
Medidas de similitud entre sucesiones ordenadas de grados cromáticos Dr. Oscar Pablo Di Liscia	31
Desarrollo de un sistema óptico para interfaces tangibles (mesa con pantalla reactiva) Ing. Emiliano Causa	45
Diseño de interface para el desarrollo de una pantalla sensible al tacto con aplicación musical Ing. Emiliano Causa	54
Técnicas de síntesis y procesamiento de sonido y su aplicación en tiempo real Lic. Matías Romero Costas	69
Técnicas de programación vinculadas a la realidad aumentada y a las interfaces tangibles DCV Tarcisio Lucas Pirotta	85

Desarrollo de un sistema óptico para interfaces tangibles (mesa con pantalla reactiva)

Emiliano Causa

Palabras claves

Interfaces tangibles, realidad aumentada, sistema de proyección con espejos

1. Introducción

El objetivo de este texto es mostrar el desarrollo de una mesa para pantalla sensible al tacto e interfaces tangibles. La idea consiste en diseñar una mesa que logre maximizar el tamaño de la pantalla, conteniendo la altura de la mesa dentro de un parámetro ergonómico. El problema surge a partir de que las pantallas para interfaces tangibles utilizan un proyector de video para retroproyectar la imagen que emite la pantalla. El proyector se coloca dentro del mueble, debido a que el común de los proyectores de video posee una relación de 1:1,4 a 1:1,8 (es decir, que para que el proyector, genere una imagen de 1 metro, requieren entre 1,5 y 1,8 metros de distancia a la superficie de proyector. Si pensamos que la pantalla debe estar en una posición horizontal, si queremos que la misma sea de 1 metro de ancho, se necesitaría que la mesa tenga entre 1,5 y 1,8 metros de alto. Obviamente una mesa con estas dimensiones no resulta práctica para el uso de una persona de altura promedio.

El siguiente capítulo expone el funcionamiento del sistema ReactiVision, el mismo está extraído de otro artículo del autor y se duplica aquí a fines de ilustrar el sistema:

1.1. ReactiVision y las interfaces tangibles (*)

ReactiVision (mtg.upf.es/reactable/) es una herramienta de software desarrollada por Sergi Jordà, Martin Kaltenbrunner, Günter Geiger y Marcos Alonso, quienes conforman el Grupo de Tecnología Musical dentro del Instituto Audiovisual en la Universidad Pompeu Fabra (Barcelona España). Esta tecnología permite reconocer patrones bitonales (llamados “fiducials”) impresos a piezas de interfaces tangibles que funcionan sobre una pantalla sensible al tacto. Esta interface tangible consiste en piezas de acrílico que se apoyan sobre una

pantalla sensible, ésta es capaz de reconocer las piezas, gracias a los patrones bitonales y generar respuestas audiovisuales en consecuencia. Los creadores, construyeron este software para desarrollar una pantalla sensible para interpretación musical en tiempo real (un instrumento de improvisación de música electrónica) llamada ReactTable.

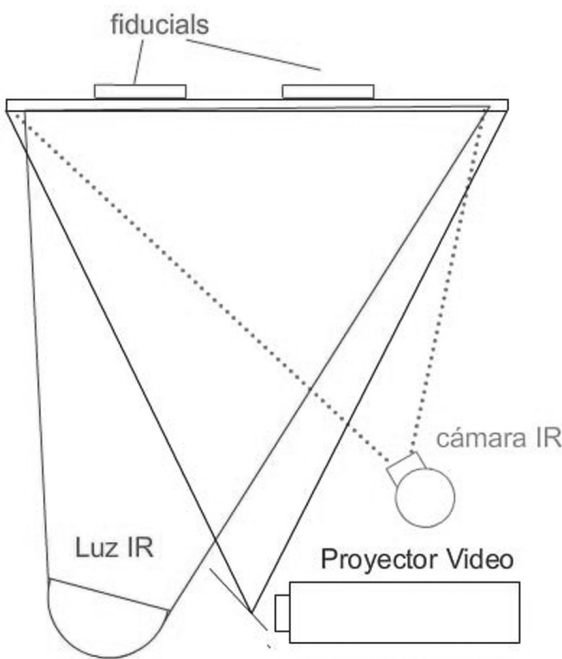


Figura: Esquema ReactTable

Como muestra la figura 3, ReactiVision permite hacer el reconocimiento de patrones bitonales, a través de un sistema óptico, que en el caso de la ReactTable se implementa con luces y cámara infrarrojas. La pantalla es un acrílico con superficie esmerilada, las imágenes se retroproyectan desde abajo usando un cañón de video, a su vez una luz infrarroja permite iluminar los patrones que serán captados por una cámara, también infrarroja. Dicha luz y cámara son infrarrojas para no interferir la luz del proyector de video (que pertenece al rango visible de la luz), y para que la cámara no vea a su vez las proyecciones.

2. Modelo inicial

Como modelo inicial se partió de un sistema de tres

espejos que utilizamos en un proyecto personal (anterior a esta investigación) y que había mostrado eficacia para reducir la altura de una mesa con una pantalla de retroproyección, el sistema en cuestión utiliza otra tecnología de captura y la superficie de proyección es una tela elástica, pero el principio de reducción de altura y ampliación del tamaño de la pantalla es el mismo.



A partir de este modelo se tomó la decisión de ponerlo a prueba con un sistema que permitiera medir su eficiencia a la hora de maximizar el tamaño de la pantalla.

3. Requisitos para el diseño de la mesa

A la hora de diseñar la mesa existen un conjunto de restricciones que la misma debería respetar:

1. La altura de la mesa no debería sobrepasar los 115 cm.
2. La utilización de espejos no debería producir deformaciones en la imagen.
3. La utilización de espejos no debería producir reflejos que dupliquen porciones de la imagen (superposiciones).
4. Los espejos deberían estar ubicados de forma tal que no proyecten sombra sobre la imagen.
5. Los espejos deberían permitir ubicar al proyector en una posición horizontal (dado que algunos proyectores no pueden ser usados en posiciones oblicuas).

Para probar el diseño se desarrolló un algoritmo que permite visualizar (modelizar) el haz de luz del proyector y sus reflejos en el sistema de espejo.

4. Sistema de tres espejos

Como se pudo apreciar en el modelo original, un sistema de tres espejos permiten resolver algunos de estos requisitos.

El espejo más pequeño permite reflejar el haz del proyector de forma tal que el dispositivo se encuentre horizontal y la imagen se emita en forma vertical.

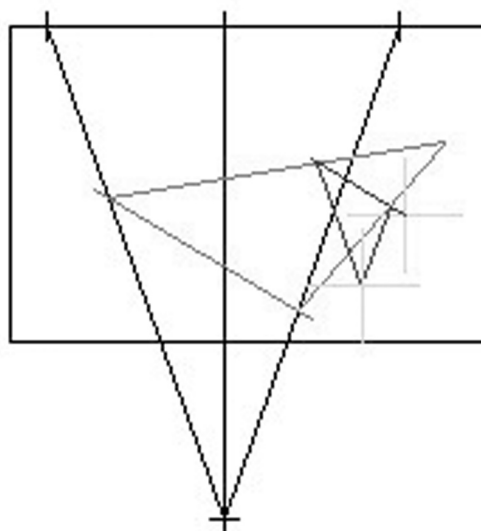
Los espejos, medio y grande permiten desviar el haz para extender la longitud del mismo (es decir, la distancia entre el proyector y la pantalla) sin extender la altura de la mesa.

A continuación se puede ver una foto del modelo original y la forma en que los espejos funcionan:



La proyección entre el espejo medio y el grande puede producir un efecto de trapecio sobre la imagen deformándola. La forma de solucionar esto es logrando que ambos espejos estén ubicados en forma paralela, es decir, con el mismo ángulo, ya que la deformación que produce el espejo medio es corregida por el grande.

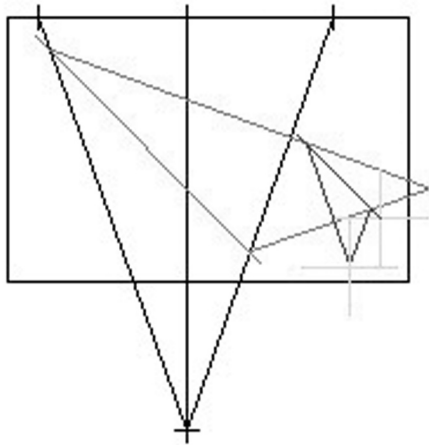
El objetivo del proyecto es hacer la mesa más pequeña posible que maximice el tamaño de la pantalla, por eso en el modelo computacional se partió de una caja de 112 cm (para dejar un margen para el apoyo en el piso) y se proyectó sobre su superficie una pantalla de 125cm de ancho. La relación entre ancho y distancia del haz se estableció en 1:1,4. Este haz se puede apreciar en el gráfico hecho por el modelo, es el triángulo de color negro. El vértice inferior corresponde a la posición en la que tendría que estar el proyector si no hubiese espejos.



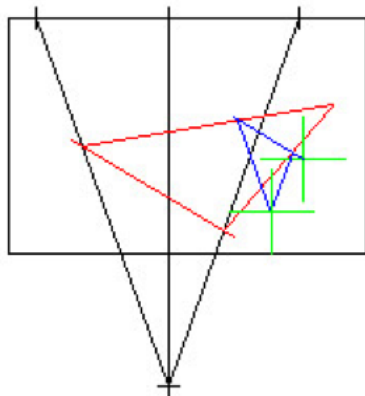
En rojo se grafica el espejo grande y la forma en que este refleja el haz. Aquí cabe aclarar que el sentido de la luz es inverso, es decir que la luz sale del proyector y viaja hacia la pantalla, pero en el modelo se parte desde la pantalla porque éste es un elemento constante (inamovible del sistema), mientras que las posiciones de los espejos y el proyector son los elementos variables del mismo.

Por último, en azul se puede ver al espejo medio y su reflejo. El vértice del haz azul muestra la posición del proyector. El espejo pequeño sirve a los fines de verticalizar el haz con el proyector puesto en forma horizontal. El tipo de modificación que produce en la longitud del haz es despreciable y no es necesario incluirlo en el sistema.

En el gráfico mostrado arriba, el ángulo de los espejos es $34,75^\circ$, esto permite que los mismos no generen sombras sobre el haz ni reflejos parásitos.



En esta nueva gráfica los espejos se ubican a $45,15^\circ$ de ángulo. Con este nuevo ángulo, los tamaños de los espejos se agrandan de a 39 a 50 cm (espejo medio) y 97 a 134 cm (espejo grande), lo que encarece el mueble y reduce el espacio para el resto del equipo, a la vez que deja menos margen de error. Esto es importante dado que no siempre se cuenta con un proyector que respete la relación 1:1,4 y el margen de error permite absorber estos cambios. También se reduce el espacio para ubicar el proyector.



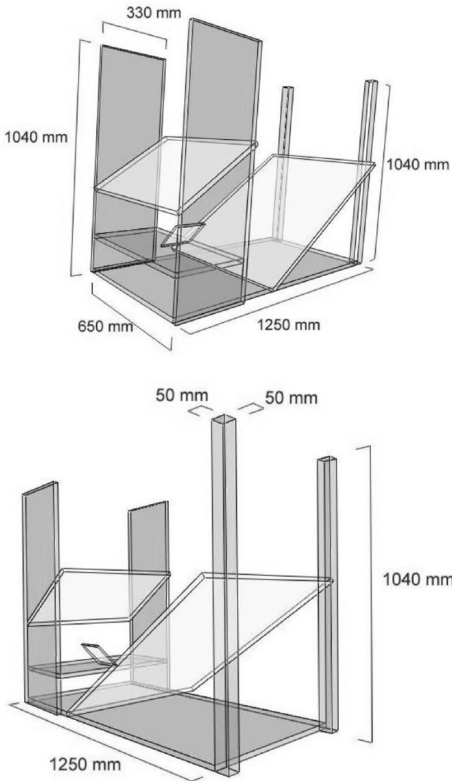
Aquí mostramos otro ejemplo en donde el ángulo se modificó a $30,57^\circ$. Como se puede ver en el gráfico, el espejo medio hace sombra en el haz que parte del espejo grande, y dada la posición es probable que el proyector también lo haga. Desde esta perspectiva, los ángulos cercanos a las 35° son buenos para un haz que sale de un proyector de relación 1:1,4.

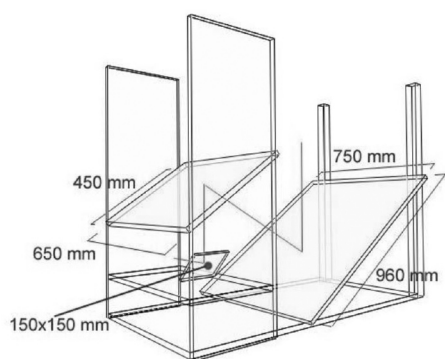
4.1. Del método

Esta forma de simulación permite probar diferentes ángulos y posiciones hasta entroncar los adecuados para cumplir con los requisitos del diseño. Es cierto que sería posible realizar un programa que busque por sí mismo dichas posiciones y ángulos haciendo el recorrido de un espacio solución, el equivalente a plantear un sistema de funciones a ser maximizadas, cuyas variables son las posiciones y los ángulos. Pero, dado que la mesa está pensada para tener cierta tolerancia respecto del tipo de proyector (específicamente, respecto de su relación ancho-profundidad), estas funciones son más complejas de ser definidas. En este punto, la complejidad y tiempo necesario para programar este algoritmo, puede ser reemplazado por un trabajo más sencillo y la utilización de ciertos criterios heurísticos, como se ha hecho en este caso.

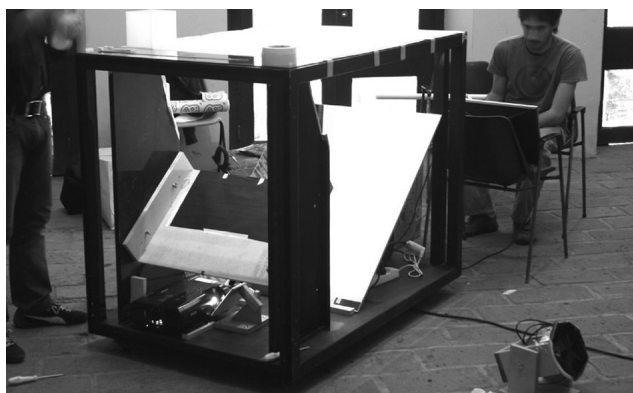
4.2. Diseño de la estructura interna

En función de los resultados obtenidos con el programa se diseñó la estructura interna del mueble. A continuación puede verse una serie de gráficos que muestra sus medidas:

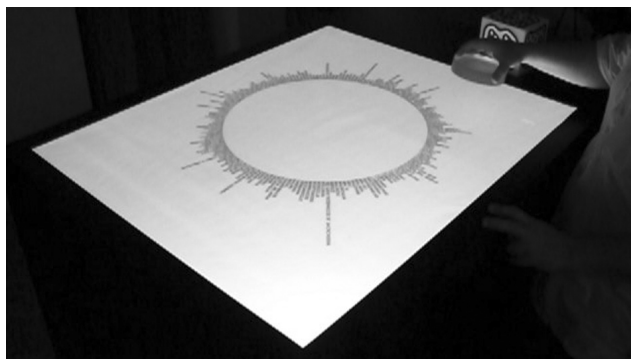




La construcción de la estructura, así como el diseño de la estructura externa del mueble estuvo a cargo de Diego Pimentel y Mariano Cataldi. Debajo se puede ver una fotografía del mueble construido antes de ser cerrado con sus tapas.



Las siguientes fotos muestran el mueble ya cerrado:



5. Deficiencias del diseño

El diseño realizado a partir del uso del programa resultó eficiente a la hora de realizar la proyección de la imagen, coincidiendo en la realidad con los resultados mostrados en el modelo computacional. Sin embargo aparecieron problemas en la implementación con aquellos elementos que no estuvieron en el modelo, particularmente con el sistema de captación.

5.1. Ubicación de la cámara

El primer problema surge a la hora de querer ubicar la cámara adecuada para poder captar la totalidad de la pantalla. El tamaño de la pantalla es de 125 cm de ancho, siendo que la cámara tiene una relación entre el ancho del cuadro y la distancia cercana a 1:1,10, la distancia a la que tendría que estar una cámara es aproximadamente de 137 cm. Obviamente la altura del mueble no alcanza y, por otra parte, la ubicación del espejo grande impide poner la cámara en esa posición, sin embargo, es posible ubicar a la cámara mirando hacia el espejo grande para extender la distancia a la pantalla.

En el complejo sistema de espejos, pensados para el proyector, las posiciones que la cámara puede usar no son las más adecuadas, obligando a obtener cuadros torcidos y deformados de la pantalla. Este defecto, por suerte es corregible con el sistema de calibración que utiliza ReactiVision.

5.2. Resolución del sistema

Otra paradoja que surge del agrandamiento de la pantalla es que el tamaño que deben alcanzar los patrones (fiducials) es mayor, ya que la resolución de la cámara no alcanza para captar patrones pequeños. Esto no es en sí una deficiencia del diseño, ya que es corregible mediante el uso de cámaras de mayor calidad.

5.3. La superficie de la pantalla y los reflejos

Uno de los mayores focos de problema está en el material que se elige para usar de superficie de retroproyección (este tema es ampliamente tratado en el texto de Diego Pimentel). Esta superficie debe cumplir con una serie de requisitos complejos de lograr:

1. Resistencia: debe ser lo suficientemente resistente para permitir apoyar objetos y posar los dedos para interactuar (a esto se suma el hecho de que la altura de la mesa invita al público a apoyarse sobre esta).
2. Opacidad: debe ser lo suficientemente opaca para permitir ver nítidamente (desde abajo) solo los objetos apoyados, pero no aquellos por encima. Por ejemplos, debe poder verse los dedos apoyados, pero no la mano por encima de ellos, ni las caras de las personas.

3. Transparencia y nitidez: también debe poder ser lo suficientemente transparente para dejar ver la imagen proyectada por detrás, pero lo suficientemente opaca como para atrapar la luz y no dejarla pasar. Por ejemplo: si es muy transparente, la proyección seguirá de largo (proyectándose en el techo, en vez de en la pantalla), si es muy opaca no podrá verse bien la imagen (hasta puede perder nitidez).

4. Antirreflejo: la superficie debajo de la pantalla debe poseer un tratamiento antirreflejo ya que el sistema de captura es muy sensible a los altos contrastes, principalmente si los producen los reflejos en esta superficie. Cuando la superficie no es antirreflejo, funciona como un espejo que refleja el sistema de iluminación.

En nuestra implementación no pudimos encontrar una superficie que cumpla con todos estos requisitos. El material que mejor cumplía el segundo y tercer requisito es el papel vegetal (de tipo calco), pero claramente no cumplía el primer requisito, lo que obligó a apoyarlo sobre un acrílico que no cumplía con el cuarto requisito.

5.4. La iluminación

La iluminación que este sistema utiliza es infrarroja. Nosotros optamos por utilizar lámparas incandescentes filtradas con acetatos de color rojo y azul, que permiten generar luz infrarroja con focos comunes. Este tipo de iluminación tiene como ventaja su bajo costo y la facilidad de encontrar esos elementos en el mercado de nuestro país. Una de las más importantes desventajas es el calor que genera, lo que obliga a tomar medidas de ventilación. El sistema alternativo que más se utiliza es el de reflectores de *leds*, que no genera calor y tienen una mayor durabilidad.

El mayor problema encontrado con la iluminación fue con la superficie de la pantalla, que al no ser antirreflejo funcionaba como un espejo más. Este pseudo espejo terminaba haciendo que la cámara vea las lámparas, lo que produce ruido en la captación de patrones. Debido a esto nos vimos obligados a poner las lámparas lo más apartadas hacia los lados posible, para que no entre en el área del reflejo, pero esta solución genera un tipo de iluminación desigual que complica nuevamente la captura.

5.5. Conclusión respecto de la deficiencias

Las deficiencias que hemos enunciado obligan a repensar en el futuro el diseño de este mueble, integrando a su evaluación estas nuevas variables que se han encontrado durante la implementación. Particularmente, la posición de la cámara, la posición y el tipo de iluminación, y el material de la superficie de proyección. Consideramos que los logros obtenidos en cuanto a la extensión del tamaño de la

pantalla no terminan de justificar algunos de los problemas encontrados, sin embargo, la mayor parte del problema se relaciona con el tipo de material, y si este factor se resolviese el balance del diseño sería positivo y simplemente habría que corregir la posición de la cámara y la iluminación, que sería sencillo de resolver.

6. Algoritmo de cálculo de reflexiones

A continuación se muestra el algoritmo utilizado para el cálculo de la reflexiones. Este se organiza a través de una serie de objetos (clases) que son: *Cono_Luz* (que es usado para modelar el haz de luz), *Segmento* (utilizados para calcular las reflexiones) y *Punto* (para establecer ciertas posiciones en el modelo).

La principal parte del algoritmo es la función “reflejar” que permite establecer la reflexión de un haz de luz en un espejo:

```
void reflejar( Segmento espejo , Segmento original
, Segmento antes , Segmento despues ){

    float anguloEspejo = angulo( espejo );
    float anguloOriginal = angulo( original );

    float largoTotal = largo( original );

    Punto contacto = obtieneCruceDosLineas( espejo.a
, espejo.b , original.a , original.b );

    antes.iniciar( original.a , contacto );
    float largoAntes = largo( antes );

    float diferencia = anguloOriginal-anguloEspejo;

    float nuevoAngulo = anguloEspejo - diferencia;
    float largoDespues = largoTotal-largoAntes;

    despues.iniciar( contacto , largoDespues , nuevoAngulo );
}
```

7. Referencias bibliográficas

<<http://www.biopus.com.ar>>.

<<http://www.mtg.upf.es/reactable/>>.

<<http://www.processing.org>>.

(*) *Este capítulo está extraído del artículo “Desarrollo de una Aplicación con Interfaces Tangibles” del mismo autor, y fue agregado*

(a pesar de la obvia duplicación) debido a la pertinencia de sus contenidos para el presente texto

8. Anexo código de algoritmo

```
Cono_Luz cono;

float mx = 100;
float my = 300;

float botton = 0;
float top = -112;
float left = 0;
float right = 170;

void setup(){

    size( 800 , 600 );
    background( 255 );
    rectMode(CORNERS);
    noFill();

    translate( mx , my );

    float anchoPantalla = 125;
    float relacion = 1.4;
    cono = new Cono_Luz( anchoPantalla , relacion );

    float profundidad = cono.profundidad;
    float margen = 13;
    float xproyector = margen + anchoPantalla/2;
    float yproyector = profundidad + top;
    cono.ubicar( xproyector , yproyector , radians(270) );

    cono.dibujar();

    float correr = 0;
    float x1esp = 107+correr;
    float y1esp = -8.5;
    float x2esp = 27;
    float y2esp = -64+correr;

    Segmento espejo = new Segmento( x1esp , y1esp , x2esp , y2esp );

    float anguloEspejo = angulo( espejo );
    float tamañoEspejo = largo( espejo );

    println( "anguloEspejo = " + (180+degrees( anguloEspejo )) );
    println( "tamañoEspejo = " + tamañoEspejo );
```

```

rect( left , top , right , botton );

Segmento antes_1 , despues_1 , original_1 ;
original_1 = new Segmento( cono.extremoR , cono.foco );
antes_1 = new Segmento();
despues_1 = new Segmento();
reflejar( espejo , original_1 , antes_1 , despues_1 );

Segmento antes_2 , despues_2 , original_2 ;
original_2 = new Segmento( cono.extremoL , cono.foco );
antes_2 = new Segmento();
despues_2 = new Segmento();
reflejar( espejo , original_2 , antes_2 , despues_2 );

stroke(255,0,0);
espejo.dibujar();
despues_1.dibujar();
despues_2.dibujar();

Punto baseEspejo2 = new Punto( 158, -43 );
float tamEspejo2 = 39;
Segmento espejo2 = new Segmento( baseEspejo2 , tamEspejo2 , anguloEspejo );

Segmento antes_3 , despues_3 , original_3 ;
antes_3 = new Segmento();
despues_3 = new Segmento();
reflejar( espejo2 , despues_1 , antes_3 , despues_3 );

Segmento antes_4 , despues_4 , original_4 ;
antes_4 = new Segmento();
despues_4 = new Segmento();
reflejar( espejo2 , despues_2 , antes_4 , despues_4 );

baseEspejo2.dibujarCruz();
stroke(0,0,255);
espejo2.dibujar();
despues_3.dibujar();
despues_4.dibujar();

despues_4.b.dibujarCruz();
despues_4.b.string();
}
void draw(){

}

//CLASE CONO DE LUZ

class Cono_Luz{

```

```

float relacion;
float anchoPantalla;
float profundidad;

float x,y,angulo;

Punto foco, centro, extremoR , extremoL;

//-----

Cono_Luz( float anchoPantalla_ , float relacion_ ){

    anchoPantalla = anchoPantalla_;
    relacion = relacion_;
    profundidad = anchoPantalla * relacion;

    ubicar(0,0,0);
    calcularCono();
}
//-----

void ubicar( float x_ , float y_ , float angulo_ ){
    x = x_;
    y = y_;
    angulo = angulo_;
    calcularCono();
}
//-----

void calcularCono(){

    foco = new Punto( x , y );
    centro = puntoUbicadoA( foco , profundidad , angulo );
    extremoR = puntoUbicadoA( centro , anchoPantalla/2 , angulo + HALF_PI );
    extremoL = puntoUbicadoA( centro , anchoPantalla/2 , angulo - HALF_PI );
}
//-----

void dibujar(){
    foco.dibujarCruz( color(0,0,0) , 5 );
    centro.dibujarCruz( color(0,0,0) , 5 );
    extremoR.dibujarCruz( color(0,0,0) , 5 );
    extremoL.dibujarCruz( color(0,0,0) , 5 );

    dibujaLinea( foco , extremoR );
    dibujaLinea( foco , extremoL );
    dibujaLinea( extremoL , extremoR );
    dibujaLinea( foco , centro );
}

```

```

    }
}

//-----
// FUNCIONES MATEMÁTICAS
//-----

float linea( float x , float x1 , float x2 , float y1 , float y2 ){
    return (x-x1)/(x2-x1)*(y2-y1)+y1;
}
//-----

float lineaLim( float x , float x1 , float x2 , float y1 , float y2 ){
    float valor = (x-x1)/(x2-x1)*(y2-y1)+y1;
    valor = min(valor,y2);
    valor = max(valor,y1);
    return valor;
}
//-----

float menorDistAngulos( float origen , float destino ){
    float distancia = destino - origen;
    return anguloRangoPI( distancia );
}
//-----

float anguloRango2PI( float angulo ){
    float este = angulo;

    for( int i=0 ; i<100 ; i++ ){
        if( este >= TWO_PI ){
            este -= TWO_PI;
        }
        else if( este < 0 ){
            este += TWO_PI;
        }
        if( este >= 0 && este <= TWO_PI ){
            break;
        }
    }
    return este;
}
//-----

float anguloRangoMenos2PI( float angulo ){
    float este = angulo;

    for( int i=0 ; i<100 ; i++ ){
        if( este > 0 ){

```

```

        este -= TWO_PI;
    }
    else if( este <= -TWO_PI ){
        este += TWO_PI;
    }
    if( este > - TWO_PI && este <= 0 ){
        break;
    }
}
return este;
}
//-----

float anguloRangoPI( float angulo ){
    float este = angulo;
    for( int i=0 ; i<100 ; i++ ){
        if( este > PI ){
            este -= TWO_PI;
        }
        else if( este <= -PI ){
            este += TWO_PI;
        }
        if( este >= -PI && este <= PI ){
            break;
        }
    }
    return este;
}
//-----
// CLASE: SEGMENTO
//-----
class Segmento{
    Punto a,b;
    //-----

    Segmento(){
        a = new Punto();
        b = new Punto();
    }
    //-----

    Segmento( Punto a_ , Punto b_ ){
        iniciar ( a_ , b_ );
    }
    //-----

    Segmento( float x1 , float y1 , float x2 , float y2 ){
        iniciar( x1 , y1 , x2 , y2 );
    }
}

```



```

//-----

Segmento( Punto a_ , float distancia , float angulo ){
    iniciar( a_ , distancia , angulo );
}
//-----

void iniciar( float x1 , float y1 , float x2 , float y2 ){
    a = new Punto( x1 , y1 );
    b = new Punto( x2 , y2 );
}
//-----

void iniciar( Punto a_ , float distancia , float angulo ){
    a = new Punto();
    a.copiarDe( a_ );
    b = puntoUbicadoA( a , distancia , angulo );
}
//-----

void iniciar ( Punto a_ , Punto b_ ){
    a = new Punto();
    b = new Punto();
    a.copiarDe( a_ );
    b.copiarDe( b_ );
}
//-----

void dibujar(){
    line( a.x , a.y , b.x , b.y );
}
}
//-----
float largo( Segmento este ){
    return dist( este.a.x , este.a.y , este.b.x , este.b.y );
}
//-----
float angulo( Segmento este ){
    return atan2( este.b.y - este.a.y , este.b.x - este.a.x );
}
//-----
void reflejar( Segmento espejo , Segmento original , Segmento antes , Segmento
despues ){

    float anguloEspejo = angulo( espejo );
    float anguloOriginal = angulo( original );

    float largoTotal = largo( original );

```

```

    Punto contacto = obtieneCruceDosLineas( espejo.a , espejo.b , original.a ,
original.b );

    antes.iniciar( original.a , contacto );
    float largoAntes = largo( antes );

    float diferencia = anguloOriginal-anguloEspejo;

    float nuevoAngulo = anguloEspejo - diferencia;
    float largoDespues = largoTotal-largoAntes;

    despues.iniciar( contacto , largoDespues , nuevoAngulo );
}
//-----
// CLASE: PUNTO
//-----
class Punto{
    float x,y;
    //-----

    Punto(){
        iniciar( 0 , 0 );
    }
    //-----

    Punto( float x_ , float y_ ){
        iniciar( x_ , y_ );
    }
    //-----

    void iniciar( float x_ , float y_ ){
        x = x_;
        y = y_;
    }
    //-----

    void copiarDe( Punto otro ){
        x = otro.x;
        y = otro.y;
    }
    //-----

    void string(){
        println( "(  "+x+"  ;  "+y+"  )" );
    }
    //-----

    void dibujarCruz( ){
        dibujarCruz( color(0,255,0) , 20 );
    }
    //-----

```

```

void dibujarCirculo( color esteCol , float ancho ){
    stroke( esteCol );
    ellipse( x , y , ancho*2 , ancho*2 );
}
//-----

void dibujarCruz( color esteCol , float ancho ){
    stroke( esteCol );
    line( x-ancho , y , x+ancho , y );
    line( x , y-ancho , x , y+ancho );
}

}
//-----
Punto puntoUbicadoA( Punto a , float distancia , float angulo ){
    Punto aux = new Punto();

    float xx = a.x + distancia * cos( angulo );
    float yy = a.y + distancia * sin( angulo );

    aux.iniciar( xx , yy );

    return aux;
}
//-----
float distancia( Punto a , Punto b ){
    return dist( a.x , a.y , b.x , b.y );
}
//-----
void dibujaLinea( Punto a , Punto b ){
    line( a.x , a.y , b.x , b.y );
}
//-----
Punto obtieneCruceDosLineas( Punto p1 , Punto p2 , Punto p3 , Punto p4 ){
    Punto aux = new Punto();

    float x = 0;
    float y = 0;

    float a = p2.y - p1.y;
    float b = p2.x - p1.x;
    float c = p4.y - p3.y;
    float d = p4.x - p3.x;

    if( b != 0 && d != 0 ){

        float ab = a/b;
        float cd = c/d;

```

```

float e = p3.y - p1.y + p1.x*ab - p3.x*cd;

float p = ab-cd;

if( p != 0){
    x = e / p;
    y = linea( x , p1.x , p2.x , p1.y , p2.y );
}

}
else{
    if( b == 0 && d != 0 ){
        x = p1.x;
        y = linea( x , p3.x , p4.x , p3.y , p4.y );
    }
    else if( b != 0 && d == 0 ){
        x = p3.x;
        y = linea( x , p1.x , p2.x , p1.y , p2.y );
    }
}

aux.iniciar(x,y);

return aux;
}

```